SecureUnionID介绍

一、背景

在广告业务中，为了更好地进行广告投放的决策，需求方平台（Demand-Side Platform，DSP）往往需要对从媒体方接收到的用户设备标识符（Device Identity，DID）与自己数据库中的DID进行匹配，来查看该DID是否已经在自己数据库中，如果匹配成功则可以查看之前的参与竞价的记录以及用户画像来更好地进行决策。举个简单的例子来解释这个场景，阿里巴巴提供了一个DSP用来进行广告投放，当一个用户登录媒体方字节旗下抖音App时会有一些广告位来让多个DSP竞价，这时app会将用户的设备DID发送给包括阿里巴巴在内的DSP，这些DSP会查询自己的数据库，如果存在数据库中，则会根据该之前的历史行为来竞价，如果不存在，则会按照之前设定的默认竞价策略来竞价，同时存储这个DID以及对应的竞价信息，用于后续竞价和广告投放，收到竞价后，抖音app会根据最高的竞价选择DSP，让这个DSP来决定此广告位投放哪个广告。对于上述流程，当前主要采取的方法主要有两种：

1） 媒体方直接发送明文DID。

2） 媒体方发送DID的哈希值。

二、问题

上述两种方法都有明显的缺陷，

- 首先媒体方的DID刻画了一个公司的用户群体，这属于公司的机密信息，同时DID除了在广告业务用，也在其他的场景用，比如银行借贷行为，网络习惯等。 如果这个DID泄漏了，别人可以根据这个DID与用户画像关联起来，这样就知道给用户的行为了，因此DID也属于用户隐私。媒体方肯定不希望除了本身之外的实体（其他媒体和DSP）获取其DID明文，因此直接发送明文DID会导致公司的机密信息泄露。
- 其次发送DID的哈希值尽管不会暴露DID明文，但是由于哈希函数的算法都是公开的，而自己研发新的哈希函数成本过大，并且DID的构成是按照一定标准生成的，其熵值很小，因此DSP很容易地可以通过DID的哈希值猜测出DID的明文。

综上，现有技术都不能很好地保护DID。

三、设计的初衷与目标

现有方案面临的安全问题主要来源于DID没有被加密便被上传了，但是如果简单的通过让每个参与方通过自己的密钥进行加密，那么相同的DID就会得到不同的密文，这时DSP便不能够完成DID的匹配。因此，我们考虑让所有参与的媒体方参与DID的加密，由于不能直接将DID明文给其他媒体方，需

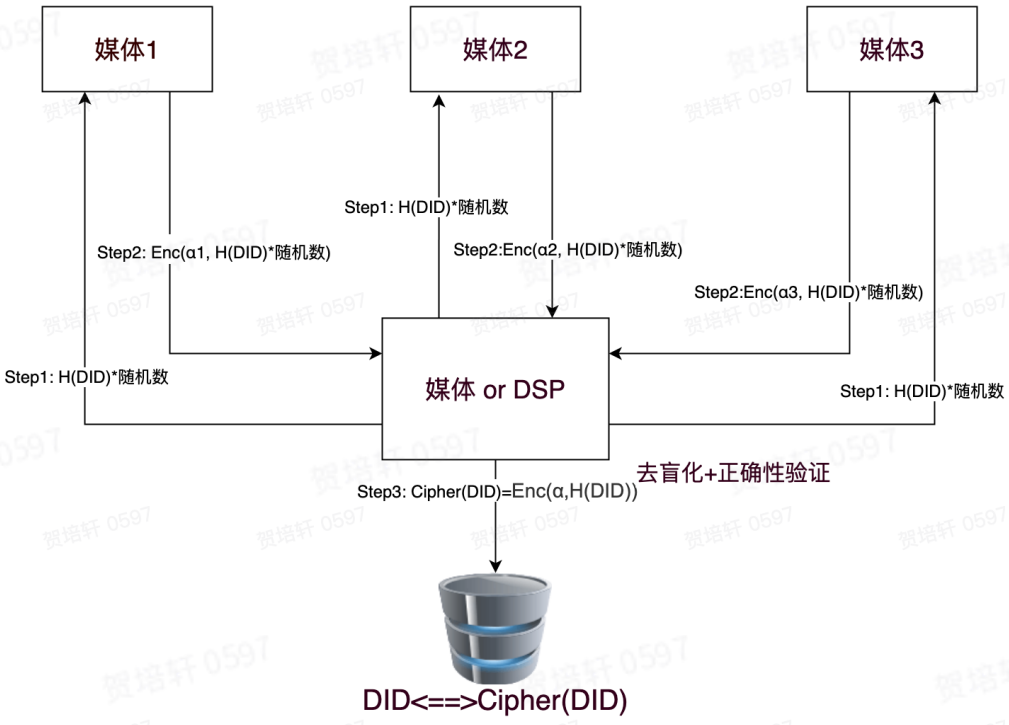
要先让请求加密方把DID进行哈希后乘上随机数进行盲化，然后让其他媒体方对这个盲化后的消息进行加密返回给请求加密方，这时请求加密方就可以合并密文并进行去盲化得到最终由所有媒体共同加密DID。

倘若每个媒体方都按照上述流程走，那么对应相同的DID就会有相同的密文，这时DSP就可以实现DID的匹配，由于上述流程可以并行的进行，因此对于系统的性能影响不大。需要注意的是，由于密文的生成是由互不信任的其他媒体方参与的，因此我们的方案实现了让请求加密方拥有能够验证收到密文的正确性的能力。对于该方案，期望实现的目标如下：

1. **可用性**：在保护DID明文不泄露的前提下能够让DSP与媒体方完成DID的匹配，同时不影响太多的性能。
2. **数据机密性**：任何一个没有该DID明文的DSP都无法从密文得知其明文。
3. **抗恶意媒体攻击**：任何一个在系统作假的媒体方都会被发现。

四、方案介绍

各个媒体和DSP都需要在离线阶段针对自己数据库中的DID明文让所有媒体方进行加密，然后得到一张<DID,Cipher(DID)>的对应表，其中Cipher(DID)表示DID的密文。 之后在在线阶段，媒体就只传加密的DID，DSP收到之后查自己数据库中的<DID,Cipher(DID)>进行匹配。下图是本方案主体部分（与其他媒体方沟通得到DID密文）的一个简单的流程图，大致可以分成以下几个步骤：



步骤1：媒体方或者DSP对数据库中所有的DID进行哈希操作，然后乘上随机数进行盲化，将盲化后的结果发送给其他所有媒体方。

步骤2：其他媒体方接收到消息后，对该消息用自己的私钥αi加密该消息并返回。

步骤3：媒体方或者DSP将合并这些加密结果，并且用之前的随机数去盲化，并得到了用全体媒体方私钥加密的DID密文。同时对该密文的正确性进行验证，如果验证失败，对之前收到的加密结果以此验证找到作假的媒体方。

运行完上述步骤后，媒体方或者DSP便都可以获得<DID,Cipher(DID)>的对应表，可以方便地进行在线阶段的广告竞价。

五、方案具体算法

具体的实施方法分为初始化、离线阶段和在线阶段：

- **初始化**
  - 任意选择一个参与方来生成系统参数 $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), g_1, g_2)$ ，其中 p 是个大素数，其为乘法循环群 $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ 的阶， g_1, g_2 分别是群 $\mathbb{G}_1, \mathbb{G}_2$ 的生成元， $e(\cdot,\cdot)$ 是一个双线性映射函数。之后通知其他所有参与方。
  - 每个媒体选择一个 α_i 作为自己的私钥，并公开公钥 $PK_i = g_1^{\alpha_i}, PK_i' = g_2^{\alpha_i}$ 。
- **离线阶段**
  - 所有的媒体和DSP计算系统公钥

    $$PK = \prod_{i\in\{1,\dots,n\}} PK_i = g_1^{\alpha}, PK' = \prod_{i\in\{1,\dots,n\}} PK_i' = g_2^{\alpha} \ .$$

    这里 $\prod$ 为连乘操作，并且假设总共有 n 个媒体方。
  - 所有媒体或者DSP为了建立<DID,Cipher(DID)>的对应表，会针对所有DID进行加密。首先为每个DID选取一个随机数 $\beta \in \mathbb{Z}_p$ ，然后将DID哈希后进行盲化处理 $M = H_G(DID) \cdot g_1^{\beta}$ ，这里 $H_G(\cdot)$ 是 $\{0,1\}^* \to \mathbb{G}_1$ 的哈希函数，这样其他媒体方不能通过M猜测到DID具体是什么。
  - 作为请求方将M发送给其他所有媒体，每个媒体 j 计算加密结果 $c_j = M^{\alpha_j}$ 返回给请求方。其中 α_j 表示第 j 个媒体方的私钥。
  - 之后媒体方和DSP的操作略有不同：

    1）对于DSP来说，收到所有媒体方的加密结果后，计算最终去盲化后的加密DID

    $$Cipher(DID) = \prod_{i\in\{1,\dots,n\}} c_i \cdot PK^{-\beta} = H_G(DID)^{\alpha} \ 。$$

    2) 对于媒体方来说，收到其他所有媒体方的加密结果后，要先计算自己的加密结果 $c_{self} = M^{\alpha_{self}}$ ，然后再去计

    算最终去盲化后的加密DID。

    $$Cipher(DID) = \prod_{i\in\{1,\dots,n\}\setminus self} c_i \cdot c_{self} \cdot PK^{-\beta} = H_G(DID)^{\alpha} \ 。$$
  - 正确性的验证。判断 $e(Cipher(DID), g_2) = e(H_G(DID), g_2)^{\alpha} = e(H_G(DID), PK')$ 是否成立，若不成立则有人作假。通过判断哪个

$e(c_j \cdot PK_j^{-\beta}, g_2) = e(H_G(DID), g_2)^{\alpha_j} = e(H_G(DID), PK_j')$ 不成立就可以找到具体的造假方。这步即验证签名的合法性。

- 同时当拥有大量的针对不同的DID的加密结果 $Cipher(DID_k)$ 我们也提供批量验证，直接判断 $e(\prod Cipher(DID_k), g_2) = e(\prod H_G(DID_k), PK')$ 是否成立即可。

做完离线阶段，媒体方和DSP方都可以获得到<DID,Cipher(DID)>的对应表，具体来说，媒体方会保存明文DID和密文Ciper(DID)的对应关系；DSP方除了保存上述关系外，比媒体方多了之前针对该DID的参竞日志，其中bid_cnt表示竞标次数，show_cnt表示中标次数。

媒体侧<对应关系表>		DSP侧<对应关系表>		DSP<参竞日志>	
DID	$Cipher(DID)$	DID	$Ciper(DID)$	bid_cnt	show_cnt
DID1	Ciper(DID1)	DID1(重合)	Ciper(DID1)	2	1
DID2	Ciper(DID2)	DID2(重合)	Ciper(DID2)	0	0
DID3	Ciper(DID3)	DID3(重合)	Ciper(DID3)	1	1
DID4	Ciper(DID4)	？（非重）	Ciper(did4)	1	1
DID5	Ciper(DID5)	？（非重）	Ciper(did5)	1	0
DID6	Ciper(DID5)				
DID7	Ciper(DID7)				
DID8	Ciper(DID8)				

- **在线阶段**

当某用户登录某个媒体方的App时，该媒体方会在自己的对应关系表中查询该用户设备DIDn对应的密文 $Ciper(DIDn)$ ，然后执行如下步骤

- Step1——媒体方发送流程：针对需要竞价的DIDn，发送 $Ciper(DIDn)$ 给**DSP**。
- Step2——**DSP**的处理流程：若**对应关系表**中可能有如下三种情况。
  - **对应关系表**为 $< DIDn, Ciper(DIDn) >$ ，则 DSP明确了解DIDn是谁，根据该DID的价值进行竞价，竞价成功则使用**重定向用户广告召回逻辑**来投放广告。
  - 在 对应关系表中无法找到 $Ciper(DIDn)$ ，则根据默认策略进行竞价，并更新**对应关系表**为 $<$ 空$, Ciper(DIDn) >$ ，且更新参竞日志。

- **对应关系表**为 $< 空, Ciper(DIDn) >$ ，则 DSP不了解DIDn是谁，则根据相应价值进行竞价，若竞价成功参竞日志可以帮助频控，同时使用**非重定向用户召回广告逻辑**来投放广告。

六、总结

这里我们想要强调的是此方案的应用场景，不单单只能在所述场景中使用，其应用场景十分的广泛。我们将这个问题抽象化，其本质上为一个轻量级的多方隐私保护集合求交方案，借助于一个第三个可以方便地进行求交操作。

一个常见的场景就是，多个没能力自建私有云的单位（例如政府机构、医院等）把自己的数据加密上传到公有云上，这时公有云就行可以方便地统计多家医院交集数据，例如可以找出多家医院所有胃癌病例数据，这样有利于后续各家医院进行联合的医学研究。

另一个更具广泛应用性的场景就是联邦学习，当有多个参与方共同参与联邦学习时，可以采用我们上述的算法方便高效地完成加密样本对齐。